

Application Serial No. 10/728,730
Reply to Office Action of July 5, 2007

Amendments to the Drawings:

The attached drawing sheet(s) include(s) changes to Fig.1 . This sheet, which includes Fig.1 , replaces the original sheet.

REMARKS/ARGUMENTS

The Examiner rejects claims 1-31 on the ground of statutory obviousness-type double patenting as being unpatentable based on claims 1-11 of U.S. 6,374,261.

Independent claim 1 of the '261 patent, the only independent claim, is as follows:

1. A method of automatically updating a knowledge base comprising a database that stores information pertaining to a subject, comprising:

in response to obtaining a file of information that is at least in part expressed in natural-language form, analyzing the file by computer to identify therein types of information pertaining to the subject;

analyzing by computer the information of the identified types by executing an intelligent filter that uses heuristics to identify therein items of information for storage in the knowledge database;

extracting by computer the found items of information from the file;

arranging by computer the extracted items by their types into a database record; and

storing by computer the database record in the knowledge database.

Applicant respectfully contends that the pending claims are not obvious over this claim.

At least the following italicized features of the pending independent claims are not obvious over claim 1:

1. A system for parsing an arbitrary input stream, comprising:
a plurality of parsers operable to parse an input stream, each parser corresponding to a unique input structure;
a parser selection agent operable to receive the input stream and select a subset of the plurality of parsers to parse the input stream, wherein the input stream comprises a plurality of differing input structures and wherein the selected subset of parsers produce multiple parser outputs corresponding to the plurality of differing input structures; and
an encoding agent operable to convert the multiple parser outputs to a common grammar.

8. A method for parsing an arbitrary input stream, comprising:
- (a) receiving an input stream, the input stream comprising information defined by *at least first and second input structures*;
 - (b) providing at least a portion of the input stream to *each of a plurality of parsers, the plurality of parsers corresponding to differing sets of grammars*;
 - (c) receiving output from *each of the plurality of parsers*; and
 - (d) based on the outputs of the plurality of parsers, performing at least one of:
 - (i) selecting a first output from a first parser that corresponds to the first input structure and a second output from a second parser that corresponds to the second input structure; and
 - (ii) selecting a first parser corresponding to the first input structure to parse one or more first segments of the input stream and a second parser corresponding to the second input structure to parse one or more second segments of the input stream.

23. A method for parsing computer generated information, comprising:
- receiving a stream of information, *the stream being generated by one of a plurality of possible different computational sources, wherein each computational source generates a stream corresponding to a unique input structure and wherein each of a plurality of differently structured segments of the stream is free of an embedded tag indicating a corresponding computational source and/or input structure for the respective segment*;
 - comparing at least a portion of the stream with a set of tokens to provide a subset of tokens identified in the *at least a portion of the stream*;
 - heuristically identifying, *from among at least one of a plurality of possible input structures and a plurality of possible computational sources, at least one of an input structure corresponding to the at least a portion of the stream and a computational source for the at least a portion of the stream*; and
 - parsing the stream based on the identified at least one of an input structure and computational source.

28. An autonomous heuristic parser, comprising:
- an input operable to receive a stream of information, *the stream being generated by one of a plurality of possible different computational sources, wherein each computational source generates a stream corresponding to a unique input structure*; and
 - a parser operable to (a) compare at least a portion of the stream with a set of tokens to provide a subset of tokens identified in the *at least a portion of the stream*; (b) heuristically identify, *from among at least one of a plurality of possible input structures and a plurality of possible computational sources, at least one of an input structure corresponding to the at least a portion of the stream and a computational source for the at least a portion of the stream*; and (c) parse the stream based on the identified at least one of an input structure and computational source, wherein the parser is not provided with an input structure identifier, other than the corresponding input structure itself, either in or external to the *at least a portion of the input*

stream to identify or assist in the identification of the at least one of the respective input structure corresponding to the at least a portion of the stream and a computational source for the at least a portion of the stream.

Accordingly, the nonstatutory obviousness-type double patenting rejection should be withdrawn.

The Examiner objects to Fig. 1 as lacking a "Prior Art" legend. Fig. 1 has been amended as requested.

The Examiner objects to dependent claims 9 and 10 as being of improper dependent form. Applicant disagrees with this objection. Claim 8 requires "performing *at least one of*" substeps (i) and (ii). Thus, contrary to the Examiner's statements, Claim 8 requires that only *one or both* of substeps (i) and (ii) be performed. It does not require that *both* steps be performed. By positively requiring substep (i) to be performed, claim 9 does further limit claim 8. The same argument applies to claim 10, which positively requires substep (ii) to be performed.

The Examiner rejects claims 1-31 under 35 U.S.C. §102(e) as being anticipated by Johnson (U.S. 2002/-141449).

Applicant respectfully traverses the Examiner's rejections. Johnson fails to teach or suggest at least the following italicized features of the pending independent claims:

1. A system for parsing an arbitrary input stream, comprising:
 - a plurality of parsers operable to parse an input stream, each parser corresponding to a unique input structure;
 - a parser selection agent operable to receive the input stream and select a subset of the plurality of parsers to parse the input stream, wherein the input stream comprises a plurality of differing input structures and wherein the selected subset of parsers produce multiple parser outputs corresponding to the plurality of differing input structures; and
 - an encoding agent operable to convert the multiple parser outputs to a common grammar.*

8. A method for parsing an arbitrary input stream, comprising:
 - (a) receiving an input stream, the input stream comprising information defined by at least first and second input structures;
 - (b) providing at least a portion of the input stream to each of a plurality of parsers, the plurality of parsers corresponding to differing sets of grammars;

- (c) receiving output from each of the plurality of parsers; and
- (d) based on the outputs of the plurality of parsers, performing at least one of:
 - (i) selecting a first output from a first parser that corresponds to the first input structure and a second output from a second parser that corresponds to the second input structure; and
 - (ii) selecting a first parser corresponding to the first input structure to parse one or more first segments of the input stream and a second parser corresponding to the second input structure to parse one or more second segments of the input stream.

23. A method for parsing computer generated information, comprising:
receiving a stream of information, the stream being generated by one of a plurality of possible different computational sources, wherein each computational source generates a stream corresponding to a unique input structure and *wherein each of a plurality of differently structured segments of the stream is free of an embedded tag indicating a corresponding computational source and/or input structure for the respective segment;*
comparing at least a portion of the stream with a set of tokens to provide a subset of tokens identified in the at least a portion of the stream;
heuristically identifying, from among at least one of a plurality of possible input structures and a plurality of possible computational sources, at least one of an input structure corresponding to the at least a portion of the stream and a computational source for the at least a portion of the stream; and
parsing the stream based on the identified at least one of an input structure and computational source.

28. An autonomous heuristic parser, comprising:
an input operable to receive a stream of information, the stream being generated by one of a plurality of possible different computational sources, wherein each computational source generates a stream corresponding to a unique input structure; and
a parser operable to (a) *compare at least a portion of the stream with a set of tokens to provide a subset of tokens identified in the at least a portion of the stream;* (b) *heuristically identify, from among at least one of a plurality of possible input structures and a plurality of possible computational sources, at least one of an input structure corresponding to the at least a portion of the stream and a computational source for the at least a portion of the stream; and* (c) parse the stream based on the identified at least one of an input structure and computational source, *wherein the parser is not provided with an input structure identifier, other than the corresponding input structure itself, either in or external to the at least a portion of the input stream to identify or assist in the identification of the at least one of the respective input structure corresponding to the at least a portion of the stream and a computational source for the at least a portion of the stream.*

U.S. 2002/0141449 to Johnson

Johnson is directed to a method for parsing a bit stream including multiple *data* (not message header) formats, and an apparatus and computer program including a set of parsers and parser-selection and invocation capabilities for handling parsing of multiple data formats. A first parser is selected and invoked to handle a first formatted component of the bit stream, and this selected parser selects and invokes a next parser which is capable of handling a differently formatted next component of the bit stream.

The architecture includes a plurality of message brokers 30 to manage the flow of information between applications. The brokers route a message to multiple destinations using rules acting on the contents of one or more fields in a message or message header, transform a message so that applications using differing formats can exchange messages in their own formats, and, within a message flow, the action to be taken can be defined according to a message structure, message topic, or data within the message. The brokers do not transform messages to a common format as this would render the architecture inoperable.

Each message flowing through a message broker has a specific structure, which is important and meaningful to the applications that send or receive that message. Message structure information as used in IBM's MQSeries Integrator products comprises a message domain, message set, message type, and wire format of the message. Together these values identify the structure and format of the data the message contains. Every message flow that processes a message conforming to this structure and format must understand it to enable the message bit stream to be interpreted. (Johnson at ¶[0035]).

The message type and format information for messages predefined in the message repositories is typically included in the messages' headers, and so a parser can recognize the message structure and format when the message is received. (Id. at ¶[0037]; see also ¶¶[0044] and [0119]). The message structures that can be handled by the message brokers include those which are predefined within one or more message repositories 60.

A parser selector 80 of the broker 30 *sends the Message Descriptor or MD component* to an MD parser 50, which accesses a structure template for MDs which is stored in the message repository 60 and applies this to the received MD to model it as a sequence of ordered name-value pairs [*id.* at ¶[0044]). The MD parser then reads one or more predefined message fields identifying the next message component's type and/or its format, compares this information with a list of component types/formats and selects and invokes another parser 50 which has a predefined responsibility for handling parsing of components having this type and/or format. The MD parser also specifies what portion of the bit stream it has consumed to indicate where a next selected parser 50 should begin. If the next component is the RFH2 header, then this component is given to a specific RFH2 parser. The RFH2 parser applies a stored RFH2 template to parse the header, modelling it as a sequence of name-value pairs, and then this RFH2 parser reads one or more fields relating to the next component. If the next component is the XML data portion, then the RFH2 parser invokes an XML parser in response to identifying that the component comprises XML. Since the templates stored for the MD and message headers determine what is the last field within their respective message component, the MD and RFH2 parsers can easily determine when to invoke the next parser.

Once the relevant parsers within the set of available parsers have been invoked to handle their respective chunks of the message bit stream and to create a syntax element tree, this syntax element tree can then be manipulated by the message broker's processing nodes.

From the foregoing, a number of observations can be made. First, Johnson does not teach sending a signal segment to multiple parsers, serially or in parallel, and determining which of the parsed outputs most likely corresponds to the correctly parsed output and selecting the responsible parser as the correct parser for additional signal segments (claims 8-10 and 14). In Johnson, the parser is selected by the parser selector 80 *before* parsing is performed based on message type and format information in the message header. Each parser, and not the parser selector, then selects the next parser to be invoked for subsequent portions of the message. Second, Johnson does not teach comparing the input stream with a set of tokens to provide a

subset of identified tokens and using the set of identified tokens to identify possible computational source(s) and/or input structure (claims 12, 23-24, 28, and 29). Third, Johnson does not teach an encoding agent to transform parsed output from multiple parsers corresponding to differing sets of grammar or grammar rules into a common grammar (claims 1-2, 6-7, 18, and 19). Rather, the message broker converts messages into differing formats depending on the message recipient. Fourth, Johnson teaches away from the use of nonstandardized message headers (claims 3, 11, 24, and 25). Johnson requires a standardized header in the message flows. This is so because the header is relied upon for identifying the message type and format. Fifth, Johnson does not teach a chain of unique responsibility parser architecture in which an error message is returned to the client not only when no parser can parse a received string but also when multiple parsers can parse the received string (claims 5 and 13). A non-error condition exists when a unique parser can parse the received string. Sixth, Johnson does not teach an autonomous heuristic parser that can parse data from multiple sources using differing language dialects without being informed in advance the source or dialect (or without receiving a tag as part of the input string), commonly using a declarative programming rather than a procedural programming approach (claims 4, 12, 15, 23, 25, 26, 28 and 31). The heuristic parser can be configured as a higher level object parser selection agent, which decides from a portion of an input string which parser is a match and thereafter provides the entire input stream to the selected lower level object parser. The heuristic parser compares the input stream to a group of tokens and grammars, identifies one or more of the tokens and grammars that match a parsed portion of the input stream and, based on the identified set of tokens and grammars, identifies the appropriate input structure from a number of possible input structures to use in parsing the input string.

Accordingly, the claims are allowable.

The dependent claims provide added reasons for allowability.

By way of example, Johnson does not teach or suggest determining and assigning to a selected input stream segment a set of flags corresponding to a set of values depending on the

Application Serial No. 10/728,730
Reply to Office Action of July 5, 2007

presence or absence of a syntactical and/or semantical relationship. The flags are used to identify at least one of an input structure and a computational source for the segment. (See claims 25 and 30.)

Dependent claims 16-19 are directed to recursive analysis of a parse tree as part of converting parsed output to a common grammar.

Based upon the foregoing, Applicants believe that all pending claims are in condition for allowance and such disposition is respectfully requested. In the event that a telephone conversation would further prosecution and/or expedite allowance, the Examiner is invited to contact the undersigned.

Respectfully submitted,

SHERIDAN ROSS P.C.

By: 

Douglas W. Swartz
Registration No. 37,739
1560 Broadway, Suite 1200
Denver, Colorado 80202-5141
(303) 863-9700

Date: Oct 4, 2007